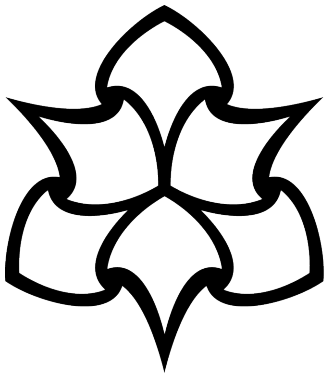


An implementation-focussed bio/algorithmic workflow for synthetic biology

Angel Goñi-Moreno†, Marta Carcajona†, Juhyun Kim†, Esteban Martínez-García†, Martyn Amos‡ and Vitor de Lorenzo*†

† Systems Biology Program, Centro Nacional de Biotecnología, Cantoblanco-Madrid, Spain.
‡ Informatics Research Centre, Manchester Metropolitan University, United Kingdom.

E-mail: vdlorenzo@cnb.csic.es
*To whom correspondence should be addressed

Abstract

As synthetic biology moves away from “trial and error" and embraces more formal processes, workflows have emerged that extend from the conceptualisation of a genetic device to its construction and measurement. We are particularly interested in this latter aspect (i.e., characterisation and measurement of synthetic genetic devices), as this is a workflow component that has received relatively little attention, but is crucial to the success of such constructions. We present an end-to-end use case for engineering a basic synthetic construct, which is supported by information standards and computational methods, and which focuses on characterisation and measurement. This workflow captures the main stages of genetic circuit design and description, and offers standardised tools for both population-based measurement and single-cell analysis. The main contributions of the current paper are (1) Consideration of specific *vector features*. Although circuit design has been successfully automated, important structural information is usually overlooked, as is the case of plasmid vectors. We advocate the use of the Standard European Vector Architecture to select the optimal carrier for a design and a thorough description, in order to unequivocally correlate digital definitions and molecular devices. We developed a digital version of this plasmid format with the Synthetic Biology Open Language and a software

tool that allows the user to embed genetic parts in vector cargoes. This enables the annotation of a mathematical model of the circuit's kinetic reactions formatted with the Systems Biology Markup Language. From that point onwards the experimental results and their *in silico* counterparts proceed alongside, with constant feedback to preserve consistency between them; (2) A framework for the *calibration* of fluorescence-based measurements in synthetic biology. One of the hardest endeavours in standardisation, metrology, is addressed by reinterpreting the experimental output in light of simulation results, allowing us to turn arbitrary fluorescent units into relative measurements; (3) Integration of single-cell methods into a framework for *multicellular* simulation and measurement, allowing for standardised consideration of the interplay between the carrier chassis and culture conditions.

Introduction

Synthetic biology is concerned with the rational design and construction of biological information processing devices.[1] The rigorous application of *engineering principles and processes* is fundamental to the success of this endeavour,[2,3,4] and significant attention is now being paid to the development of standardised *workflows*,[5,6] which describe sequences of biological and algorithmic processes required to obtain a desired outcome. Such workflows, therefore, specify a " tool-chain" for synthetic biology, and the anticipated benefits of using them include *modularity* (allowing individual processes to be implemented in several different ways), *robustness* and *scalability* .

One of the over-arching challenges for the field is the end-to-end *automation* of "biodesign",[7,8] a process that is made up of two main stages:[6] (1) the automatic selection and/or construction of biological components, and their assembly into a network that, in principle, performs information processing according to a high-level specification, and (2) the fine-tuning of the system components and/or architecture to obtain the desired performance. The first part of this process concerns the detailed *specification* of the components to be used [9,10] (or fabricated [11,12,13]), the attendant data representation and storage issues,[14] and the correct arrangement of components into a *circuit* that can implement a given (logical) function. A wealth of so-called "bio-CAD" tools now exist for this latter task,[15,16] and these include SBROME,[17,18]

TinkerCell,[19] and SynBioSS.[20] In terms of "fine-tuning" (the second stage), recent developments use post-assembly modification of constructs based on observed network behaviour[6] or the "evolution" of cell models,[21] facilitating an iterative "homing in" approach towards circuit design.

In this paper, we focus on the latter stages of the circuit engineering process (that is, the implementation stages that follow the *initial* development of a circuit design). The *specific issues* that we address with our workflow (and, therefore, the most significant contributions of the paper) are (1) the formalisation of circuit description, (2) the effect of plasmid vectors on circuit performance, and (3) the correlation of experimental observations with simulation results. We now briefly discuss each of these.

The first stage in the post-design process is to formalise the descriptions and sequences of the parts of the system to be constructed. An early technical standard for the description of biological parts was the BioBrick,[22,23,] which is appropriate for the assembly of DNA segments. However, a key consideration (which is not handled particularly well by early standards) is the variety of *plasmid vectors* that are available for the delivery of biological parts. Importantly, the choice of plasmid vector can dramatically affect the performance of an engineered circuit; plasmid features such as replication origin, selection markers and expression system need to be carefully selected.[24]

As computational tools to aid biodesign become more commonplace, we may begin to see more uniformity in terms of the types of circuit we see in the literature. However, once they are built, the process of *measuring* the behaviour of the designed system (in order to assess its fidelity to the desired output) may still vary substantially, since few existing workflows consider measurement, and teams are free to choose their own tools for this stage. Mathematical and computational modelling have become fundamental tools in synthetic biology, but they are only effective when combined with useful *in vivo* observations of synthetic systems. In this workflow, we describe a methodology for easily mapping simulation results onto laboratory measurements.

Results and discussion

Our overall workflow is depicted in 1. We use a combined experimental *in vitro* / *in silico* approach, the two perspectives being tightly coupled at key points. The various stages are temporally ordered, from left to right, and we begin once a circuit design is established (that is, we do not consider issues of circuit design, and instead focus on implementation and measurement). The first stage in our workflow is *Description*, in which the design of the desired construct is captured by some representation(s); this then feeds into the *Implementation* stage, in which the construct is built (or modelled). Once the device has been implemented, we perform *Population-level measurement* in order to obtain aggregate performance metrics; this then feeds into a *second Implementation* phase, which facilitates closer (single-cell) observations. We now describe each workflow stage in more detail.

Description

In order to obtain reliable and robust circuit performance, it is important to have control over the vector, and to be able to compare its performance with the same plasmid in multiple scenarios. In order to achieve this, we use (for the *in vivo* component) the Standard European Vector Architecture (SEVA)25 , which is a standard for the physical assembly of vector plasmids and their nomenclature, as well as an online database of functional sequences and constructs available to the community. Paired with the SEVA description of the *plasmid* is a digital representation of the *circuit* for the *in silico* component of the workflow, for which we use the Synthetic Biology Open Language (SBOL).26 This provides a “standard exchange format" for synthetic biology designs (between research groups, and between different toolkits).

Implementation

In the first *in vitro* Implementation phase, the vector is assembled using standard molecular biology procedures, resulting in the synthesis of circuit modules, and their insertion into the carrier plasmid. In parallel with this process (i.e., during the *in silico* implementation phase), we construct a standardised digital description using SBOL, with one SBOL document per

construction (File S2). These documents are then combined (using a Java-based tool, Tool S1), resulting in a single SBOL file containing the sequences of interest in the correct cargo position according to the restriction enzyme sequences. This tool identifies those SBOL components in common across components (i.e., the restriction sites) and replaces all the information that exists in the cargo section from enzyme to enzyme with the cassette of interest. After this step, both the plasmid containing the circuit and its representation are fully standardised.

Measurements.

The use of mathematical modelling and computational analysis has became a fundamental part of synthetic biology, due to the information they provide concerning the mechanical behaviour of the systems. However, this potential can only be used effectively when combined with direct *in vivo* measurements.[27] Advances in metrology and measuring techniques will obviously benefit the field of synthetic biology. Recently, attempts have been made to standardise these. Relative Promoter Units (RPU)[28] have emerged as a measuring standard for promoter activity based on a comparison against a reference promoter. On a more abstract level, the Polymerase Operations Per Second (PoPs) measure[9] is used as the signal carrier in transcriptional circuits. However, none of these methods are free of controversy.[15]

In order to simulate the model constructed in the Implementation phase, we use the iBioSim[29] tool; conveniently, iBioSim exports reactions to a single Systems Biology Markup Language (SBML)[30] file (File S3), which is a computational standard for the representation of biochemical networks. Importantly, this allows us to link up the SBML *biochemical model* of the circuit with the SBOL description of the *DNA components* of the circuit, using the methodology described in.[31] In turn, this connects (via SBOL) with the SEVA description of the vector, giving seamless integration of information across different standards that are used for different levels of description. We also develop an application (Tool S2, based on libSBML[32]) to convert a given SBML file into Python coded scripts, used for for deterministic and stochastic simulations (File S4). Importantly, the SBML model details (i.e., rates) correspond not only to the circuit itself, but also its carrier vector. This significantly reduces output variability; by including details of the vector in the model characterisation (via SEVA/SBML) we take into consideration the possibility

that the carrier plasmid might later change, due to decisions taken at the implementation phase. Any such change will, in turn, inevitably (although, sometimes subtly) affect the observable behaviour of the model when implemented, so including details of the vector allows us to factor in fluctuations due to variable plasmid selection.

The inclusion of an extra step within the workflow for *multicellular* analysis will also help to reduce variability caused by both the chassis and culture conditions. Indeed, both chassis (i.e., *P. putida* vs. *E. coli*) and culture conditions add their own effects to the circuit and its carrier. If the circuit has to be used under different scenarios we should quantify cellular behaviour. In the example provided there are behaviours that cannot be measured with the cytometer (i.e., noise inheritance or cell movement), and which require time-lapse microscopy in order to be quantified. The parameters corresponding to these behaviours are therefore fitted according to single-cell measurements. Again, this information adds value to a potential specification sheet that accompanies the *in vivo system*.

We use spectophotometry to measure the fluorescent signal of the entire cell population; dividing this by the optical density (OD) over time yields the average fluorescence value per cell in the culture. Experimental values are used to fit kinetic rate parameters in the mathematical models so they produce similar profiles. Importantly, in the graphs that follow, the Y-axis refers to *arbitrary units* of fluorescence in experimental observations, and the *number of molecules* (of, for example, mCherry proteins) in the simulated observations. Matching the latter with the former gives us an important reference point concerning measurements, which allows us to interpret subsequent results.

We perform stochastic analysis in order to characterise noise in the system, using the well established Gillespie algorithm.[33] On the experimental front, we obtain data on noise using flow cytometry, which allows the user to check the fluorescence intensity value of (in principle) every single cell in the bacterial culture. Although the ready-to-use graphs produced by the cytometer (Figure S1) are used as standard in most laboratories, we prefer to use the *raw values*, before they are processed for presentation (normally in a “black box" fashion, which is opaque to the user). There are three main reasons for using raw cytometry data: (1) Cytometers "count" cells using

variable intervals of fluorescence at high values of a logarithmic scale that is not always constant, and which depends on a specific machine set-up. This processing therefore introduces variability that is hidden from the user; (2) We need cell-specific values in order to make direct comparisons with simulated cells within our framework; (3) Raw data values are more amenable to importing and processing by various tool-chain components, whereas the automated extraction of specific values from graphs produced by cytometers introduces unnecessary complications and the possibility of misreading data.

A simulated cytometry graph is obtained by running the Python version of the reactions (see Methods). This offers two potential benefits: firstly, it gives a computational method (via an SBML model) of discarding invalid values from the raw cytometry information (see the later Case study for an example). And, secondly, by overlapping both experimental and simulated plots we are able to correlate the arbitrary units (*au*) of the cytometer with those from the spectrophotometer. We propose this procedure as one approach towards unifying machine-based measurements in the laboratory (and give an example in the Case study, below).

Implementation (2)

The behaviour of our circuit will inevitably be affected by the specific attributes of the host cell. A thorough characterisation of a device should, therefore, include information about the performance of the *chassis*[34] (which, in our case, is *P.putida* KT2440[35]). Rather than simply providing “added value", this information is of *vital importance* in the case of multicellular applications,[36,37] which are becoming increasingly important as cell-to-cell communications are increasingly well-understood and customised.[38,39]

In order to study the behaviour of circuits *in vivo*, we use DiSCUS,[40] which is an agent-based simulation package we have previously developed to study bacterial growth. Importantly, this platform considers *physical forces* between rod-shaped bacteria, and is applicable to a wide range of organisms. This tool uses the previously generated Python scripts for the intra-cellular genetic network that is implemented by our cells. The SBML model is therefore embedded into the cellular objects of the agent-based simulator. It is important to note that there is a standard,

currently under development, called the Multi-Cellular Data Standard (MultiCellDS, http://multicellds.org/), which aims to create a data standard for sharing multicellular experimental, simulation, and clinical data. Hopefully, when released, it will facilitate sharing of configuration parameters for a specific chassis performance.

Concurrently, we prepare a 2-dimensional culture on an agarose pad,[41] and let the cells grow on a monolayer in order to facilitate visualisation in the microscope.

Single-cell measurements.

We first calibrate the *movement and the growth* of the simulated cells according to experimental observations. We monitor the successive positions of a specific cell until division, and then follow the displacement of its daughters during their lifetime(s). We then match these results against the equivalent information obtained from the simulations, and adjust DiSCUS parameters to fit the experiments. In short (see Methods for more details), this information yields the most relevant features to prioritise in DiSCUS in order to reproduce the movement of our cells *in vivo* (in the Case study, below, we give specific examples).

Spatial measurements.

After characterising the dynamics of the chassis that host our circuit, we measure its performance in a spatial scenario. We measure the fluorescence intensity of our device *in vivo*, and obtain a pixel-based image analysis of the specific colour (in our example, red) captured by the microscope. In our analysis, we translate the scale bar of the analysis into values proportional to those used in the mathematical model (see Methods for details of this conversion). As a consequence, a simulation run with the system's equations inside DiSCUS bodies, can be directly compared against experiments in regard to circuit function.

Case study

In this Section we present the results of a combined *in vivo/in silico* case study, in which we construct a simple device using our workflow. We start with a simple "always-on" source; that is,

a constitutive expression cassette. Although this device is relatively simple, compared to existing synthetic genetic constructs (such as the oscillator), we emphasise that the main focus of the current paper lies with the *measurement* of such devices. That is, the complexity of the device to be constructed is less significant for the purposes of this work, as we are concerned only with handling its *output*. Fluorescence measurements are taken in fundamentally the same way, regardless of the size or complexity of a synthetic device; what interests us here is how we might *standardise* such metrics, and relate them back to *in silico* studies in a useful and meaningful way.

The two subcomponents of the circuit are (1) the *pEM7* constitutive promoter, and (2) the red fluorescence reporter gene *mCherry* (see Methods for details - File S1). Once the initial design is in place we move to the Description stage, where the system *pEM7-mCherry* is digitally formalised and physically built. The SEVA vector pSEVA 231 (Figure 2B) is selected to carry the design. This contains a Kanamycin marker (labelled 2), origin of replication pBBR1 (labelled 3), and the default cargo sector (labelled 1). As the cargo sector is a sequence of restriction sites, we need to select specific locations into which to *paste* our modules. As depicted in Figure 2A, we complete the promoter component by flanking the sequences of restriction sites PacI and AvrII, and using HindIII and SpeI for the reporter gene (this leaves empty space in between for future usage). Once the Description phase is complete, we move to Implementation.

In Figure 3A we highlight the kinetic rates involved and the Ordinary Differential Equations (ODEs) that govern the continuous functioning of our always-on device. After cloning, Figure 4A shows the results for average fluorescence value per cell in the culture, along with deterministic simulation runs (based on the ODEs) for both the SBML model (implemented using iBioSim) and its corresponding Python script. We then move to the Population measurement phase.

Figure 4B shows the fluctuations in molecular levels of the reactions of Figure 3A when running the Gillespie algorithm on the SBML model (iBioSim) and its corresponding Python file. As expected, the observed variability is the same in both, as the kinetic rates remain unchanged (i.e., the same as in the ODEs). The mean value is precisely situated on the steady state value of the deterministic simulation.

Raw data from the cytometer are plotted on Figure 4C, where the bimodal curve tells that approximately half of the cells display strong fluorescence, while the rest express none (or very little). The latter group corresponds to invalid values, and can be discarded, as indicated by the control data (the same strain without the plasmid) and the already processed graph (Figure S1). Moreover, further microscope tests show strong fluorescence in all the cells with a relatively narrow noise interval, which confirms the correct elimination of that non-expressing cell group. As described in the workflow description, this gives a computationally standard way of discarding invalid values from raw cytometry information. Moreover, it yields a method for correlating outputs from different pieces of laboratory equipment. We illustrate this in the graph of Figure 4C; we are able to correlate the arbitrary units (*au*) of the cytometer with those from the spectrophotometer: 1 *au* in the former, and 1.2 *au* in the latter (see Methods for more information). After performing population-level measurements, we move to single-cell measurements.

5A shows the result of experiments to track cell movements. 5B shows the positions of a cell (from Figure 5A) until division, and then the displacement of its daughters during their lifetime. 5C shows the most relevant features we need to add in DiSCUS in order to reproduce the movement of our cells, starting from a very simple growth algorithm (which returns unrealistic patterns) (Figure 5C.1). Ultimately, we find that we need to include: (1) *cell size variations* (due to conditional growth), (2) variation in *transversal angles* after division, (3) *randomised directions of movement*, and (4) slight *attraction between cells* (in order to avoid the appearance of holes within the colony).

Figure 5D compares the synchrony of growth within experimental and simulated cells, yielding suggestions as to how to uncouple growth events. These graphs show the length of each cell in the population over *time* (in the laboratory experiments) or *iterations* (in the simulation). Starting with just two cells (the same setup as in 5A) which grow and divide at the same time, we observe that, after the second division (eight cells in total), the length of the cells is no longer synchronised.

We then consider the spatial scenario. Figure 6A shows the results of measuring the fluorescence

intensity of our *pEM7-mCherry* device inside the KT2440 strain, and a pixel-based image analysis of the red colour captured by the microscope. As stated above, the scale bar of the analysis is translated into values proportional to those in the mathematical model of 4b. A simulation run in DiSCUS, using the system's equations (6B (left)) can be directly compared against experiments. We verify, for instance, that daughter cells share output levels as they directly *copy* their mother's circuit at a given time (6B), and the fact that cells with slower growth tend to display a stronger light signal (due to the accumulation of fluorescence proteins).

Discussion and conclusions

Arriving at a fully standardised workflow that allow for robust and reproducible constructs will benefit synthetic biology. We describe procedures used in our lab to build and measure synthetic devices, explaining both computational and experimental investigations via a simple use case. Many recent efforts focus on a specific step depending on application interests. That is the case of automated circuit design,[5,16,18] mathematical modelling,[20] single-cell analysis,[41] metrology,[28] data representation[26] or post-construction modification.[6] Indeed, there is significant room for improvement in each step along the workflow. However, instead of focussing on a single technique, we showed how to make use of *several* of them in an end-to-end workflow, concentrating on output measurements. There are recent reviews of other workflows,[15] but these tend to focus on enumeration rather than application of techniques. Apart from the *didactic* contribution of this paper, we provide new materials needed for linking standards, such as the tool to merge SBOL documents for SEVA description, or the scripts to translate SBML into Python. In this way, we provide a useful initial workflow for newcomers to the field, as well as (more generally) a standard workflow for robust programmable biology.

Materials and Methods

Strains and plasmids. The strain used was *Pseudomonas putida* KT2440,[35] the wild-type strain derived from mt-2[42] strain cured of the TOL plasmid pWW0. The carrier plasmid for our circuit was pSEVA 231 (2: Kanamycin resistance; 3: pBBR1 origin of replication; 1: default cargo)

selected from the SEVA database (http://seva.cnb.csic.es/). We then inserted the promoter pEM7 with *PacI/AvrII* and the mCherry reporter with *HindIII/SpeI*. The final plasmid was renamed pSEVA 237R-pEM7 (already available in the database). Importantly, the sequences of interest (target circuit) were edited to remove any restriction site that the SEVA standard uses as structural elements.

SBOL-SEVA description. The SEVA format is highly structured in unambiguous functional sectors, as shown in 2B. We described, using SBOL-2.0 (specifications on the website, http://sbolstandard.org/), the SEVA vector 231 (Figure S2). The previous existing description of this vector using GenBank format43 is then improved by adding missing features (like assembly scars) and establishing structural and functional links. Separately, we produced two more SBOL documents, one for each component of the circuit. Ultimately, we developed a Java based application that can be fed with the carrier plasmid and the cassettes that needs to be inserted, and outputs the new vector. The application searches in the carrier file for those restriction sites present in the cassettes (iteratively) and substitutes the sequence in between. The resulting SBOL document has all location parameters (i.e. *bioStart*) updated.

Mathematical modelling and SBML-to-Python conversion. In the model of Figure 3A, *P* we show the promoter-reporter pair (18 copies, as estimated by previous observations for pBBR1 origin of replication44), *mRNA* the messenger RNA and *rfp* the red fluorescent protein (both at 0 molecules at the beginning of the simulation). Regarding the kinetic rates: k_1 is the transcription rate (27/18 hour^{-1}, k_2 represents the translation rate (2.5 hour^{-1}) and k_3 (0.65 hour^{-1}) and k_4 (0.265 hour^{-1}) the degradation rates of the mRNA and the protein respectively. For such a small network, parameter assignment is a difficult task due to the restricted number of constraints. Efforts on *assigning numbers* to rates45 are of vital importance at this stage.

We then used the software iBioSim (http://www.async.ece.utah.edu/iBioSim/) to write the model in SBML format and run the simulations with the Hierarchical Runge-Kutta method for ODEs solution, and the Gillespie algorithm for stochastic behaviour. The model was exported in a *flat* (iBioSim option) XML file and converted into Python scripts with the tool provided (Tool S2). Flow cytometry data was obtained from the FCS files without processing, and the simulated

graph was obtained by (1) sampling a stochastic run in time (equal time intervals), and (2) counting intensity values over a long enough (≈ 600 hours) period.

By making the simulations match experimental plots in Figure 4A, we conclude that □400 simulated molecules (s.m.) correspond to ≈400 arbitrary units in the spectrophotometer (a.u.s). As the computational measurements (s.m.) in the stochastic simulation are exactly the same, we used them to correlate the fluorescent units of the cytometer (a.u.c). As Figure 4C shows, ≈400 s.m. = ≈330 a.u.c; so 1 a.u.s = 400/330 a.u.c. We assume that the sources of fluorescent signal are the same, as the cells are unaltered.

Two-dimensional *in-vivo* setup. In order to prepare of the microscope sample, we used an agarose pad following the method described in.46 A slide glass with an attached gene frame (1.7 X 2.8 cm, life technologies) was prepared. Then 500 ul of LB, including 2% agarose, which is melted in the medium, was added into the middle of the gene frame and assembled with another slide glass. After 30 min at room temperature, one of the slide glasses was carefully removed, maintaining an intact agarose pad. Then, the pad was cut out to 5 mm width within the gene frame using a razor blade. Two strips of the pad were left to grow bacterial cells.

The strain carrying pSEVA 237R-pEM7 was precultured overnight in LB medium at 37□C and bacterial cultures were then diluted 100-fold in the same medium and grown to the exponential phase (OD600 = 0.2). 2.5 ul of the samples were then spotted on to the agarose pad and assembled with cover glasses (24 X 50 mm) for following microscopy analysis.

The widefield fluorescent microscope was used to observe the sample (Leica DMI6000B, Leica Microsystems) with a digital CCD camera Orca-R2 (Hamamatsu). The cell growth was monitored for 75 min under the microscope at 37° C and images were captured every 3 min with a40.0x/0.75 NA dry objective or 63.0x/1.3 NA glycerol immersion objective (depending on the experiment) with a bandpass filter for mCherry (BP 560/40 and EM 645/75.) using the LAS AF v. 2.6.0 software (Leica Microsystems). Images were analyzed with the MATLAB-based code Schnitzcells47 in order to track both the positions of the cells and their length while growing.

Two-dimensional *in-silico* setup. DiSCUS (http://code.google.com/p/discus/) is an agent-based software for bacterial growth that uses Pymunk (http://pymunk.readthedocs.org/en/latest/), a 2D physics library, to resolve collisions among cells. In the most basic test of Figure 5C.1 each cell is a body of 16x30 square lattice that grows lengthwise until division, when the cell is cut in half. Pressure-based growth is simulated by counting the cells that push a body of interest (threshold at 4 cells) and slowing down the growth events (without stopping them). Random angle variations were introduced after division, whereby the daughter cells *copy* the angle of the mother and add a number in the interval (-25,25) degrees. Furthermore, angle variations were included at the normal growth events, although to a smaller extent (maximum variation of 5 degrees). The fact that the cells grow *in vivo* forming a circular group without holes was simulated using a slight gravity-like value that pushed the cells towards the middle of the population. This force can be eliminated when the population is about 20 cells big, at which point the circular shape is conserved without any other attraction. Further analysis on this force is needed.

Regarding pixel intensity in the analysis of Figure 6A, we set the maximum value to be at the same level as the highest peak of the stochastic simulation of Figure 4B or the cytometry data of Figure 4C. Therefore we calculated the percentage rate (≈470*100 divided by maximum pixel value) to convert the intensity of every pixel into the scale shown by experiments. Again, we assume that the source of light is the same (KT2440) and variances are due to different machine measurements.

Acknowledgement

References

(1) Church, G. M.; Elowitz, M. B.; Smolke, C. D.; Voigt, C. A.; Weiss, R. (2014) Realizing the potential of Synthetic Biology. *Nature Reviews Molecular Cell Biology*, *15*, 289–294.

(2) Andrianantoandro, E.; Basu, S.; Karig, D. K.; Weiss, R. (2006) Synthetic biology: new engineering rules for an emerging discipline. *Molecular Systems Biology*, *2:1*.

(3) Heinemann, M.; Panke, S. (2006). Synthetic biology - putting engineering into biology.

Bioinformatics, *22*, 2790–9.

(4) Kitney, R.; Freemont, P. (2012) Synthetic biology - the state of play. *FEBS Letters*, *586*, 2029–2036.

(5) Beal, J.; Weiss, R.; Densmore, D.; Adler, A.; Appleton, E.; Babb, J.; Bhatia, S.; Davidsohn, N.; Haddock, T.; Loyall, J.; Schantz, R.; Vasilev, V.; Yaman, F. (2012) An end-to-end workflow for engineering of biological networks from high-level specifications. *ACS Synthetic Biology*, *1*, 317–331.

(6) Litcofsky, K. D.; Afeyan, R. B.; Krom, R. J.; Khalil, A. S.; Collins, J. J. (2012) Iterative plug-and-play methodology for constructing and modifying synthetic gene networks. *Nature Methods*, *9*, 1077–1080.

(7) Brophy, J. A.; Voigt, C. A. (2014) Principles of genetic circuit design. *Nature Methods*, *11*, 508–520.

(8) Densmore, D. M.; Bhatia, S. (2014) Bio-design automation: software + biology + robots. *Trends in Biotechnology*, *32*, 111–113.

(9) Baker, D.; Church, G.; Collins, J.; Endy, D.; Jacobson, J.; Keasling, J.; Modrich, P.; Smolke, C.; Weiss, R. (2006) Engineering life: building a fab for biology. *Scientific American*, *294*, 44–51.

(10) Varadarajan, P. A.; Del Vecchio, D. (2009) Design and characterization of a three-terminal transcriptional device through polymerase per second. *NanoBioscience, IEEE Transactions on*, *8*, 281–289.

(11) Nielsen, A. A.; Segall-Shapiro, T. H.; Voigt, C. A. (2013) Advances in genetic circuit design: novel biochemistries, deep part mining, and precision gene expression. *Current Opinion in Chemical Biology, 17*, 878–892.

(12) Khalil, A. S.; Lu, T. K.; Bashor, C. J.; Ramirez, C. L.; Pyenson, N. C.; Joung, J. K.; Collins, J. J. (2012) A synthetic biology framework for programming eukaryotic transcription functions. *Cell*, *150*, 647–658.

(13) Villalobos, A.; Ness, J. E.; Gustafsson, C.; Minshull, J.; Govindarajan, S. (2006) Gene Designer: a synthetic biology tool for constructing artificial DNA segments. *BMC Bioinformatics*, *7*, 285.

(14) Canton, B.; Labno, A.; Endy, D. (2008) Refinement and standardization of synthetic

biological parts and devices. *Nature Biotechnology*, *26*, 787–793.

(15) MacDonald, J. T.; Barnes, C.; Kitney, R. I.; Freemont, P. S.; Stan, G.-B. V. (2011) Computational design approaches and tools for synthetic biology. *Integrative Biology*, *3*, 97–108.

(16) Marchisio, M. A.; Stelling, J. (2009) Computational design tools for synthetic biology. *Current Opinion in Biotechnology*, *20*, 479–485.

(17) Huynh, L.; Tsoukalas, A.; Köppe, M.; Tagkopoulos, I. (2013) SBROME: a scalable optimization and module matching framework for automated biosystems design. *ACS Synthetic Biology*, *2*, 263–273.

(18) Huynh, L.; Tagkopoulos, I. (2014) Optimal part and module selection for synthetic gene circuit design automation. *ACS Synthetic Biology*, *3*, 556–564.

(19) Chandran, D.; Bergmann, F. T.; Sauro, H. M. and others. (2009) TinkerCell: modular CAD tool for synthetic biology. *Journal of Biological Engineering*, *3*, 19.

(20) Hill, A. D.; Tomshine, J. R.; Weeding, E. M.; Sotiropoulos, V.; Kaznessis, Y. N. (2008) SynBioSS: the synthetic biology modeling suite. *Bioinformatics*, *24*, 2551–2553.

(21) Cao, H.; Romero-Campero, F. J.; Heeb, S.; Cámara, M.; Krasnogor, N. (2010) Evolving cell models for systems and synthetic biology. *Systems and Synthetic Biology*, *4*, 55–84.

(22) Knight, T. (2003) *Idempotent vector design for standard assembly of biobricks*; DTIC Document.

(23) Shetty, R. P.; Endy, D.; Knight Jr, T. F. (2008) Engineering BioBrick vectors from BioBrick parts. *Journal of Biological Engineering*, *2*, 1–12.

(24) Preston, A. (2003) Choosing a cloning vector. *E. coli Plasmid Vectors*; Springer, 2003; pp 19–26.

(25) Martínez-García, E.; Aparicio, T.; Goñi-Moreno, A.; Fraile, S.; de Lorenzo, V. (2014) SEVA 2.0: an update of the Standard European Vector Architecture for de-/re-construction of bacterial functionalities. *Nucleic Acids Research*, gku1114.

(26) Galdzicki, M.; Clancy, K. P.; Oberortner, E.; Pocock, M.; Quinn, J. Y.; Rodriguez, C. A.; Roehner, N.; Wilson, M. L.; Adam, L.; Anderson, J. C. and others (2014) The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nature Biotechnology*, *32*, 545–550.

(27) Kelwick, R.; MacDonald, J. T.; Webb, A. J.; Freemont, P. (2014) Developments in the tools and methodologies of synthetic biology. *Frontiers in Bioengineering and Biotechnology*, *2*.

(28) Kelly, J. R.; Rubin, A. J.; Davis, J. H.; Ajo-Franklin, C. M.; Cumbers, J.; Czar, M. J.; de Mora, K.; Glieberman, A. L.; Monie, D. D.; Endy, D. (2009) Measuring the activity of BioBrick promoters using an in vivo reference standard. *Journal of Biological Engineering*, *3*, 4.

(29) Myers, C. J.; Barker, N.; Jones, K.; Kuwahara, H.; Madsen, C.; Nguyen, N.-P. D. (2009) iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics*, *25*, 2848–2849.

(30) Hucka, M.; Finney, A.; Sauro, H. M.; Bolouri, H.; Doyle, J. C.; Kitano, H.; Arkin, A. P.; Bornstein, B. J.; Bray, D.; Cornish-Bowden, A. and others. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, *19*, 524–531.

(31) Roehner, N.; Myers, C. J. (2013) A methodology to annotate systems biology markup language models with the synthetic biology open language. *ACS Synthetic Biology 3*, 57–66.

(32) Bornstein, B. J.; Keating, S. M.; Jouraku, A.; Hucka, M. (2008) LibSBML: an API library for SBML *Bioinformatics*, *24*, 880–881.

(33) Gillespie, D. T. (1976) A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics 22*, 403–434.

(34) Danchin, A. (2012) Scaling up synthetic biology: do not forget the chassis. *FEBS Letters 586*, 2129–2137.

(35) Nelson, K.; Weinel, C.; Paulsen, I.; Dodson, R.; Hilbert, H.; Martins dos Santos, V.; Fouts, D.; Gill, S.; Pop, M.; Holmes, M. and others. (2002) Complete genome sequence and comparative analysis of the metabolically versatile Pseudomonas putida KT2440. *Environmental Microbiology 4*, 799–808.

(36) Amos, M. (2014) Population-based microbial computing: a third wave of synthetic biology?. *International Journal of General Systems 43*, 770–782.

(37) Macía, J.; Posas, F.; Solé, R. V. (2012) Distributed computation: the new wave of

synthetic biology devices. *Trends in Biotechnology 30*, 342–9.

(38) Tamsir, A.; Tabor, J. J.; Voigt, C. A. Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature 469*, 212–5.

(39) Goñi-Moreno, A.; Amos, M.; de la Cruz, F. (2013) Multicellular computing using conjugation for wiring. *PLOS ONE 8*, e65986.

(40) Goni-Moreno, A.; Amos, M. (2015) DiSCUS: A simulation platform for conjugation computing. *Unconventional and Natural Computation* (UCNC 2015), Auckland, New Zealand, August 31-September 4, 2015.

(41) Skinner,S.O.; Sepúlveda, L. A.; Xu, H.; Golding, I. (2013) Measuring mRNA copy number in individual Escherichia coli cells using single-molecule fluorescent in situ hybridization. *Nature Protocols 8*, 1100–1113.

(42) Worsey, M. J.; Williams, P. A. (1975) Metabolism of toluene and xylenes by Pseudomonas (putida (arvilla) mt-2: evidence for a new function of the TOL plasmid.. *Journal of Bacteriology 124*, 7–13.

(43) Benson, D. A.; Karsch-Mizrachi, I.; Lipman, D. J.; Ostell, J.; Rapp, B. A.; Wheeler, D. L. (2000) GenBank. *Nucleic Acids Research 28*, 15–18.

(44) Lee, T. S.; Krupa, R. A.; Zhang, F.; Hajimorad, M.; Holtz, W. J.; Prasad, N.; Lee, S. K.; Keasling, J. D. (2011) BglBrick vectors and datasheets: a synthetic biology platform for gene expression. *Journal of Biological Engineering 5*, 1–14.

(45) Ronen, M.; Rosenberg, R.; Shraiman, B. I.; Alon, U. (2002) Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics. *Proceedings of the National Academy of Sciences 99*, 10555–10560.

(46) de Jong, I. G.; Beilharz, K.; Kuipers, O. P.; Veening, J.-W. (2001) Live cell imaging of Bacillus subtilis and Streptococcus pneumoniae using automated time-lapse microscopy. *Journal of Visualized Experiments: JoVE, 53.*.

(47) Young, J. W.; Locke, J. C.; Altinok, A.; Rosenfeld, N.; Bacarian, T.; Swain, P. S.; Mjolsness, E.; Elowitz, M. B. (2012) Measuring single-cell gene expression dynamics in bacteria using fluorescence time-lapse microscopy. *Nature Protocols 7*, 80–88.

Supporting Information Legends

File S1. Sequences of promoter pEM7 and gene mCherry.

File S2. SBOL files. For a) plasmid, b) promoter and c) reporter.

Tool S1. Software tool to merge SBOL files and insert cassettes into a vector.

File S3. SBML files.

File S4. Annotated SBML file.

Tool S2. Software tool to convert a SBML model into a Python script.

File S5. Python scripts

Figure S1. Cytometry results. Graph output by cytometer after processing.

Figure Legends

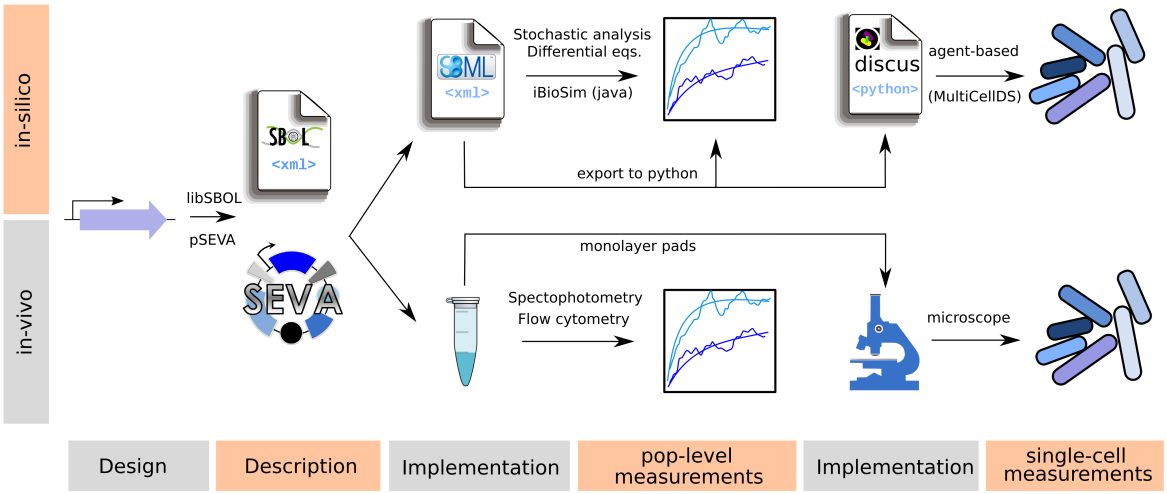


Figure 1: **Workflow for an end-to-end synthetic biology use case**. The description that follows (and modify) the design of an idea is the starting point for the consequent experimental and computational methods. The circuit and its carrier vector are described using the SEVA (Standard European Vector Architecture) format for the in-vivo workflow and the SBOL (Synthetic Biology Open Language) standard for the parallel in-silico process. A first implementation round is then performed via synthesis and cloning methods in the wet-lab and via SBML (Systems Biology Markup Language) for the modelling. The resulting material is then used for different measurements. First, we make use of usual laboratory equipment for population-based experiments (spectophotometry and flow cytometry) to compare the output against simulation software (iBioSim and *ad hoc* python code). Another implementation round prepares the samples for single-cell measurements. On the computational side, the SBML model is exported to a python script ready to be used with our software for cell movement DiSCUS (Discrete Simulation of Conjugation Using Springs). Relevant efforts are being currently done to standardise these simulations via the MultiCellDS (Multi Cellular Data Standard) project. On the other side, the cells are grown on an agarose pad for 2-dimensional populations that allow us to match results.

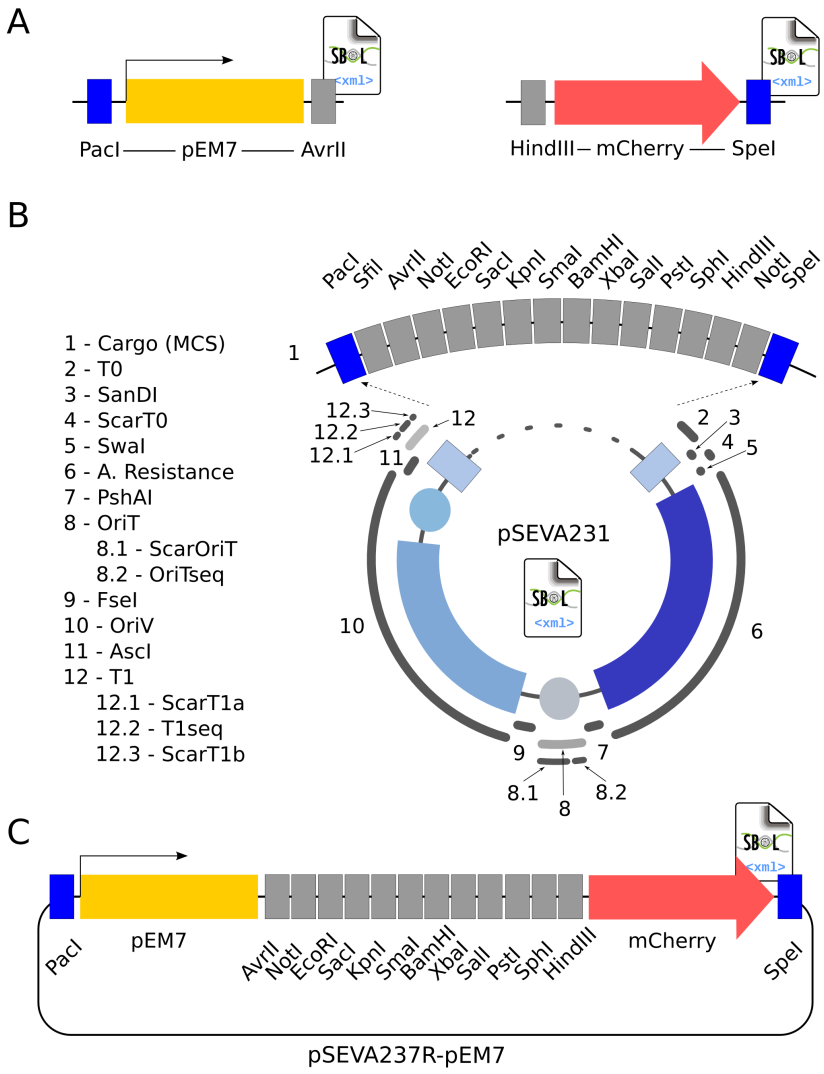
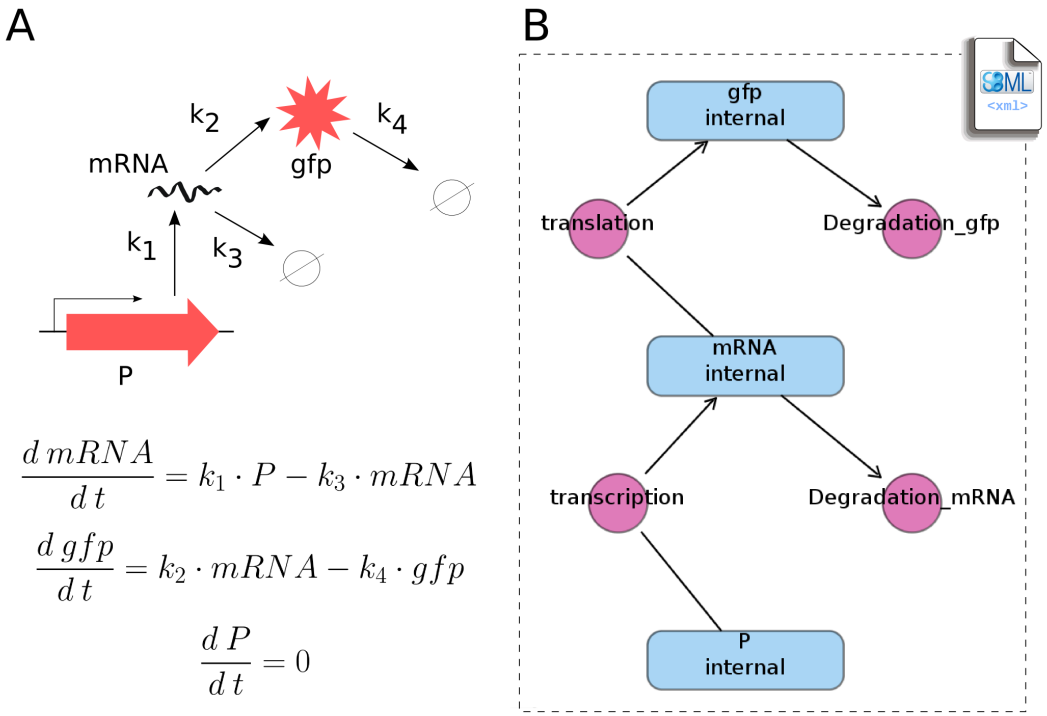
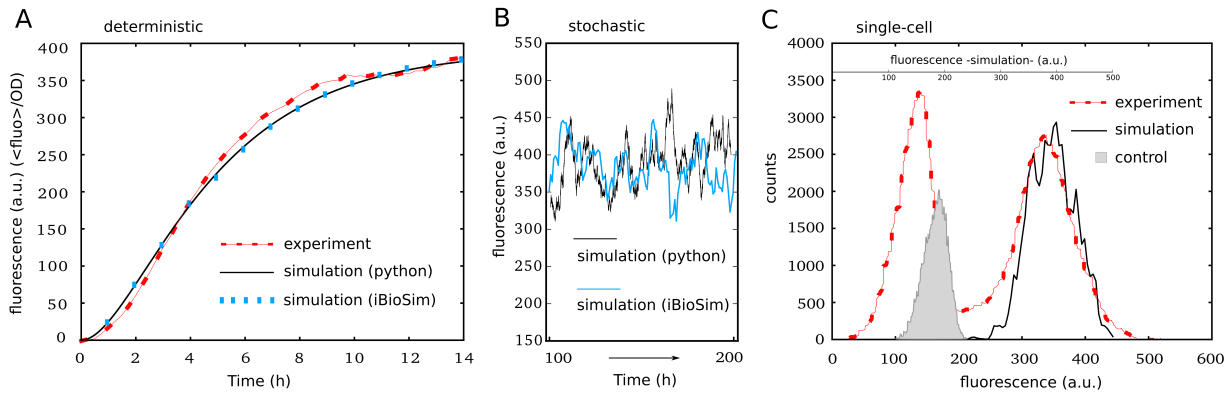


Figure 2: **SBOL description of circuit and SEVA components. A.** Circuit design modification where the components are flanked by the selected restriction sites that specify their situation inside the SEVA vector. The constitutive promoter pEM7 is surrounded by PacI and AvrII whereas the reporter mCherry is bordered by HindIII and SpeI. An SBOL document per component is created. **B.** The selected SEVA plasmid to harbour our circuit is SEVA number 231 (2: Kanamycin resistance; 3: pBBR1 origin of replication; 1: default cargo). All vector features are recorded in a single SBOL document, including cargo (multiple cloning site) components for a further assembling of circuit parts. **C.** Both in-vivo and in-silico protocols for building the final construct have the same basics: introduce, sequentially, circuit parts in the carrier vector. A software tool (Tool S1) allows to do so with SBOL documents.

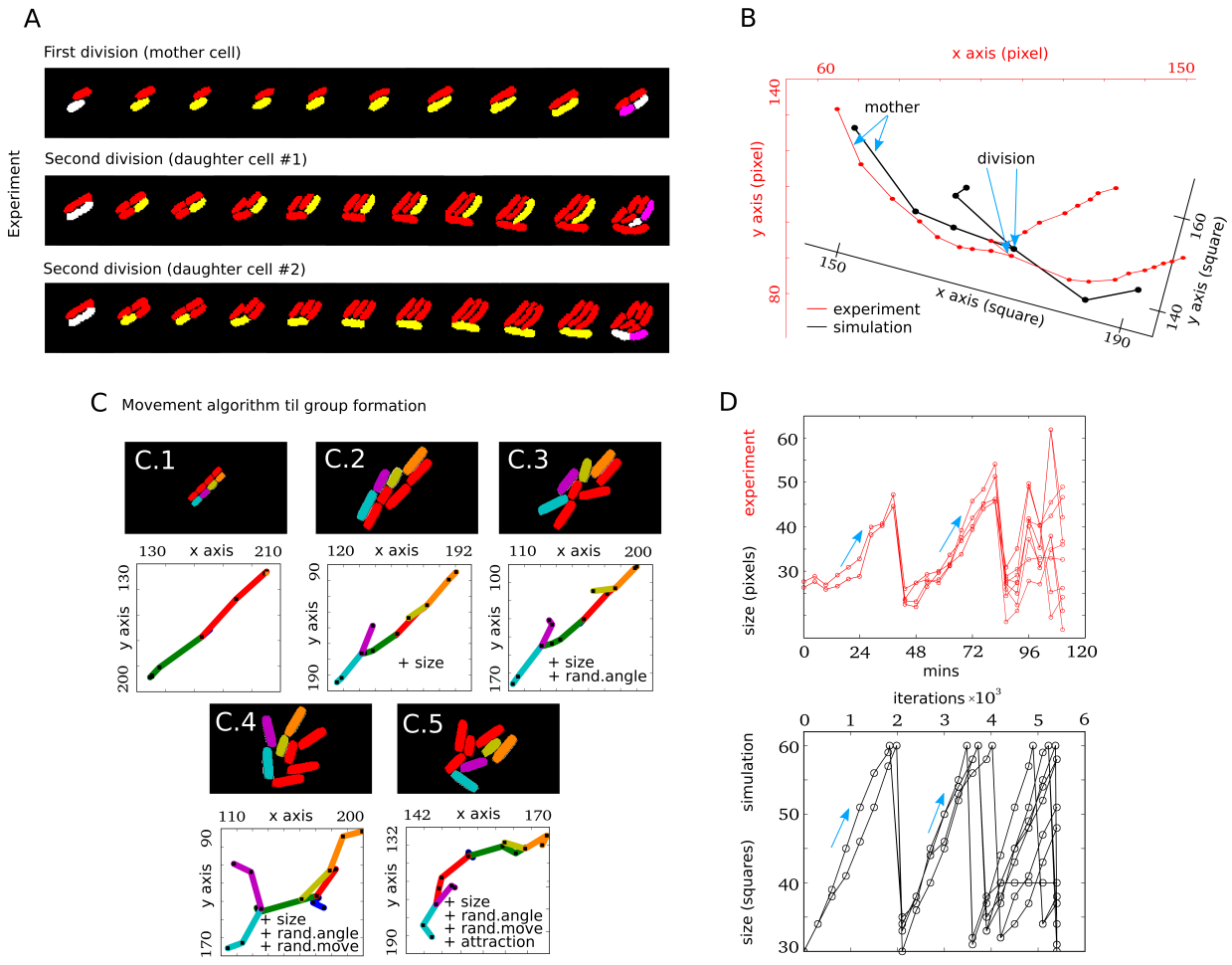


532

533 Figure3: **Mathematical modelling and its SBML format A.** Kinetic reactions (up) and system's
534 differential equations (bottom). The circuit's behaviour can be effectively simulated with just
535 four kinetic constants: the constitutive promoter *P* facilitates reporter transcription with rate k_1,
536 resulting mRNA is translated with rate k_2 leading to the formation of RFP (red fluorescent
537 protein) and both elements are degradated with rates k_3 and k_4 respectively. ODEs (Ordinary
538 Differential Equations) governing continuous dynamics are shown. **B.** Schema of the SBML
539 model produced with the software iBioSim, a CAD (computer-aided design) package for systems
540 biology. In the screenshot, blue elements represent substrates and red circles hide reaction rates.
541 After setting the parameters, iBioSim allows the user to export the model to an XML file
542 formatted following the SBML standard.



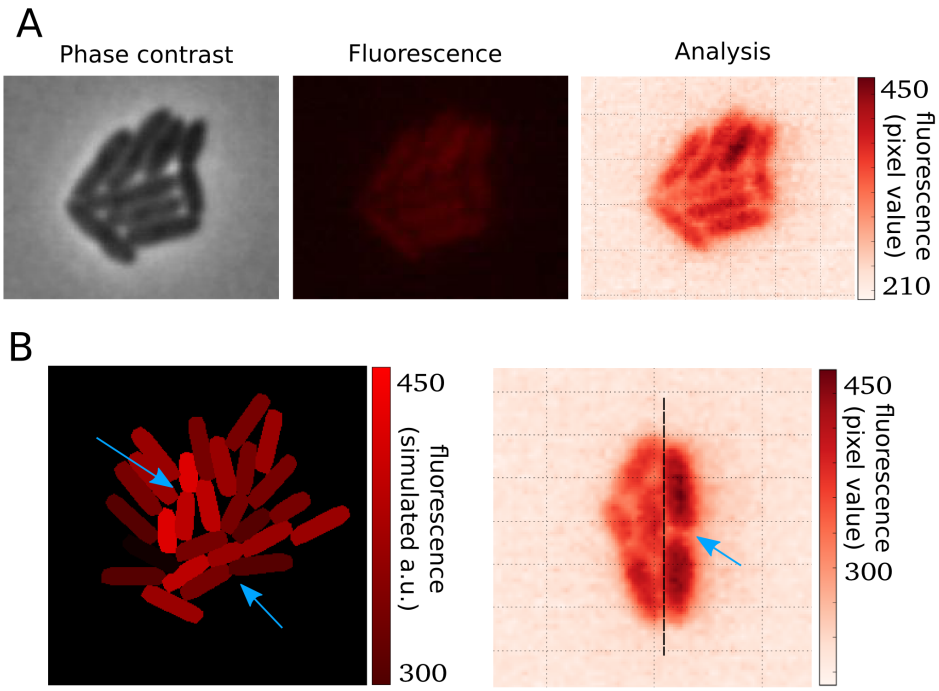
543 Figure 4: **Population-based measurements in experimental and simulation setups. A.**
544 Deterministic functioning of the circuit, in terms of fluorescence intensity over time (during 14
545 hours), averaging the value of the whole population. Red line corresponds to experimental results,
546 while blue and black lines show simulation runs of the model's differential equations with
547 iBioSim and python code respectively. Experimental values are used to fit rate numbers in
548 mathematical models so they produce similar continuous lines. **B.** Stochastic behaviour of the
549 system according to simulations. The blue line results of running the Gillespie algorithm with
550 iBioSim whereas the black line shows the python script behaviour. As expected (same algorithm
551 with equal parameters), the fluctuations are alike. **C** Fluorescence intensity values of each cell in
552 the population measures variability and expression noise. Experimental raw data extracted by
553 flow cytometry (without processing by the cytometer, see text for details) corresponds to the red
554 line. Black line results from counting expression values in the simulation with the python script,
555 while grey area represents the control (plasmid-free cells) measured experimentally. Note that
556 scales are different in simulation and experimental lines, standing for variability within arbitrary
557 units (a.u.).



558

559 Figure 5: **Characterisation of chassis mechanics. A.** Tracking cell lineages in a experimental
560 setup. Starting from the division of a single cell (up) we follow the movement of its daughters
561 (middle and bottom) in order to define their movement behaviour until next division. **B.** Position
562 coordinates are recorded during the experiment (red line) and simulation (black line) to fit
563 parameters by comparing both outputs. Cell traces are overlapped for visualisation purposes and
564 axis rotated accordingly to show dimensions. **C.** Parameter estimation for cell movement.
565 Different features are included, sequentially, in order to get the final moving procedure for in-
566 silico simulations. Starting from inaccurate movement (C.1) we add size variability due to
567 pressure (C.2), random angles after division (C.3), irregular motion changes (C.4) and slight cell
568 attraction to simulate viscous bodies (C.5). All simulations start from a single cell, and one
569 lineage is coloured to monitor coordinate positions. **D.** Synchrony of cell growth. The length of

each cell (y axis) is monitored over time (x axis) in both scenarios (experiment, up; simulation, bottom). The initial cells grow at the same time until division point is reached, whereas the third generation of cells grow asynchronously.



573 Figure 6: **Spatial progress of the genetic device. A.** Phase contrast image of population (left),
574 fluorescent picture (middle) and computational analysis (right). In the latter, the colour scheme
575 (right bar) represents the value of the red channel of every pixel from 0 to 255. However it is
576 transformed into a [0..450] scale in order to allow comparisons with previous fluorescence
577 measurements. **B.** On the left, we show a simulation of a colony starting from a single cell.
578 Upper-left arrow highlights cells with slower growth rate and RFP accumulation while bottom-
579 right arrow points at a recently divided cell where both daughters share similar RFP
580 concentration. On the right, expression noise inheritance is indicated with an arrow. Furthermore,
581 RFP accumulation caused by slow growth can be observed by the black line separation: a single
582 cell started from each side.